



Hacking y Pentesting con Python

Módulo 2: Fingerprinting y Banner Grabbing.

ÍNDICE

Módulo 2: Fingerprinting y Banner Grabbing.....	2
Presentación y objetivos	2
1 Rutina de Fingerprinting y Banner Grabbing en python.....	3
2 Integración entre metasploit framework y python	5
3 Puntos clave	9

Módulo 2: Fingerprinting y Banner Grabbing.

PRESENTACIÓN Y OBJETIVOS

En este segundo módulo se pretende explicar de qué forma se pueden detectar versiones vulnerables de servicios en ejecución empleando técnicas de Banner Grabbing y además, se podrá apreciar cómo establecer una integración completa entre Python y Metasploit Framework, algo que se utilizará en módulos posteriores y que resulta útil para automatizar técnicas de enumeración y ataque de sistemas.



Objetivos

- | Entender el procedimiento de Fingerprinting y banner grabbing para poder incorporar dichas técnicas en scripts en Python.
- | Brindar una introducción práctica al uso de la API de Sockets en Python para posteriormente, profundizar en su uso en el módulo de redes.
- | Enseñar la integración entre Python y Metasploit Framework con el objetivo de automatizar rutinas de pentesting y hacking

1 RUTINA DE FINGERPRINTING Y BANNER GRABBING EN PYTHON

Cuando se realiza una petición a un servicio en algunas ocasiones el servidor devuelve un paquete de datos que le indica al cliente que se encuentra disponible para procesar peticiones.

A veces, dicho paquete contiene información que suele ser suficiente para identificar el tipo de servicio y su correspondiente versión. Esta información es útil para un atacante a la hora de filtrar aquellos servicios que contienen vulnerabilidades conocidas y que pueden ser aprovechadas para ejecutar código arbitrario en el objetivo.

Si bien algunos servicios comunes permiten ocultar estos detalles o directamente desactivar el banner de la respuesta, es una medida poco eficiente si el servicio tiene algún tipo de vulnerabilidad, ya que solamente con interactuar con el servicio un atacante puede determinar si es explotable.

Con esto en mente, se puede crear un script que permita recorrer un segmento de red e intentar realizar una conexión con un listado de puertos definido. Por otro lado, el script puede leer un fichero de texto en donde cada línea contendrá un banner que corresponde a una versión vulnerable de un servicio y comparar cada una con el banner devuelto por el servicio analizado.

El script **2-banner_grabbing.py** recibe como argumento la IP del sistema que se desea analizar y en el bucle principal se recorren los puertos y los banners, posteriormente se comprueba si alguno coincide con el banner devuelto por el servidor y en el caso de que haya una coincidencia, se enseña por pantalla el banner, host y puerto encontrado.

El fichero de puertos, puede tener el siguiente contenido

```
21
22
25
53
79
80
110
443
8080
```

Y el fichero de banners contendrá por cada línea, un patrón que ha sido identificado como vulnerable. El siguiente listado corto representa un ejemplo valido del contenido de dicho fichero.

```
FreeBSD 8.0/7.3/7.2 i386, ProFTPD 1.3.2a/e/c Server (binary)
Debian GNU/Linux 5.0, ProFTPD 1.3.2e Server (Plesk binary)
Mandrake Linux 7.1 (apache-1.3.14-2)
OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
Samba smbd 3.X (workgroup: MYGROUP)
Cyrus pop3d 2.3.7-Invoca-RPM-2.3.7-7.el5_6.4
MiniServ 0.01 (Webmin httpd)
```

Esta forma de encontrar vulnerabilidades en los servicios que se encuentran en un segmento de red es muy simple, pero es un mecanismo bastante empleado por herramientas como Nessus o *Nexpose* para encontrar vulnerabilidades en un objetivo determinado o en un segmento de red

2 INTEGRACIÓN ENTRE METASPLOIT FRAMEWORK Y PYTHON

Metasploit Framework es una de las herramientas más conocidas para la explotación de software y labores de pentesting. Además de contar con una lista muy completa de módulos relacionados con software vulnerable de todo tipo, una de las principales características que le convierte en un framework robusto y muy flexible son las facilidades que brinda a la hora de crear exploits y módulos. El lenguaje de programación utilizado para desarrollar módulos en Metasploit Framework es Ruby, sin embargo, en Python también es posible aprovechar los beneficios que tiene este framework gracias al uso de librerías tales como `python-msfrpc`.

Antes de comenzar a explicar el funcionamiento de dicha librería, es conveniente comprender el formato `MessagePack`, el cual es utilizado por la interfaz `MSGRPC` para el intercambio de información entre cliente y servidor. `MessagePack` es un formato especializado para la serialización de la información, el cual permite que los mensajes sean mucho más compactos y livianos con el fin de transmitir información rápidamente entre diferentes máquinas.

Funciona muy similar a `JSON`, sin embargo, dado que los datos son serializados utilizando el formato `MessagePack`, el número de bytes del mensaje se reduce drásticamente. Además de lo anterior, se trata de una solución pensada para un gran número de lenguajes de programación entre los que se incluyen Python. Para conocer en mayor detalle el funcionamiento de `MessagePack` se recomienda visitar el sitio web oficial en la siguiente dirección: <http://msgpack.org/>

Dicho esto, el primer paso es utilizar el plugin **`msgRPC`** de Metasploit Framework para iniciar una instancia del servidor.

Para ello, se puede cargar el módulo desde **`msfconsole`** o directamente utilizando el comando **`msfrpcd`**

```
#!/msfconsole
msf > load msgrpc Pass=password
[*] MSGRPC Service: 127.0.0.1:55552
[*] MSGRPC Username: admin
[*] MSGRPC Password: password
[*] Successfully loaded plugin: msgrpc
msf > exit
```

también se puede utilizar la herramienta “msfrpcd” incluida en el framework

```
#!/msfrpcd -h
```

Usage: msfrpcd <options>

OPTIONS:

- P <opt> Specify the password to access msfrpcd
- S Disable SSL on the RPC socket
- U <opt> Specify the username to access msfrpcd
- a <opt> Bind to this IP address
- f Run the daemon in the foreground
- h Help banner
- n Disable database
- p <opt> Bind to this port instead of 55553
- u <opt> URI for Web server

```
#msfrpcd -P password -U admin -p 55552 -u /msfrpc -n -f
```

```
[*] MSGRPC starting on 0.0.0.0:55552 (SSL):Msg...
[*] URI: /msfrpc
[*] MSGRPC ready at 2020-03-20 01:10:24 +0100.
```

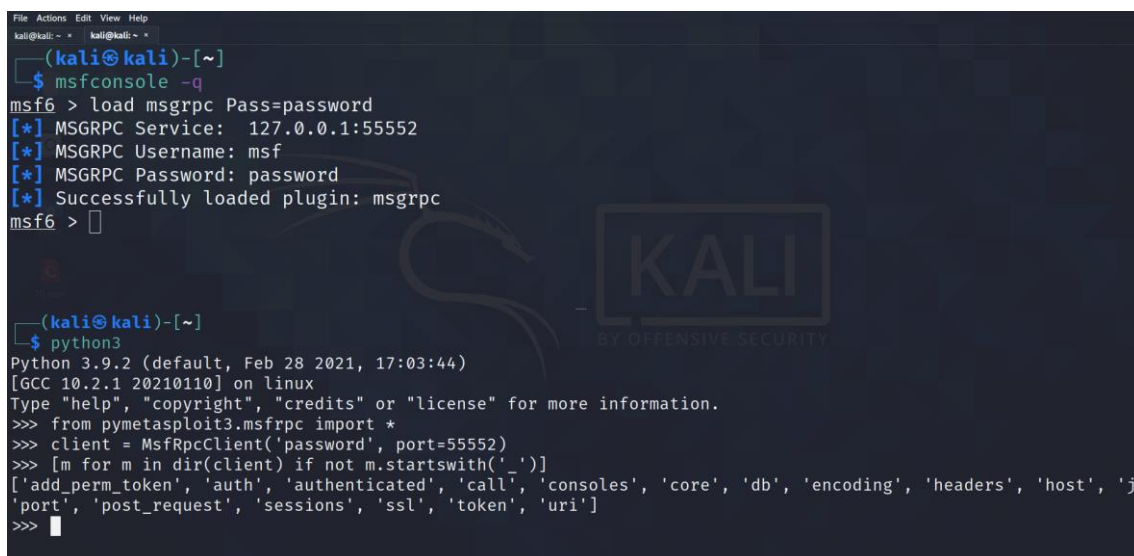
Ahora que el servicio se encuentra levantado, es el momento de instalar la librería adecuada para llevar a cabo la integración. Actualmente se cuenta con 2 alternativas: `msfrpc` (<https://github.com/SpiderLabs/msfrpc>) y `pymetasploit3` (<https://github.com/DanMcInerney/pymetasploit3>).

Cualquiera de las dos funciona correctamente a la fecha de redactar este documento, sin embargo, hay que tener en cuenta que la primera de ellas se encuentra sin actualizar desde hace varios años y el proyecto en GitHub se encuentra archivado, por ese motivo en este documento se utiliza la librería `pymetasploit3`.

Esta librería se puede instalar desde código fuente tal como se enseña en el repositorio GitHub indicado anteriormente o utilizando la herramienta PIP. Cualquiera de las dos alternativas es válida para instalar la librería.

Como se ha mencionado anteriormente, antes de empezar a utilizarla es necesario levantar el servicio MSGRPC de Metasploit Framework y a continuación establecer una conexión a dicho servicio utilizando el usuario y contraseña que se han indicado anteriormente.

La imagen que se aprecia a continuación enseña cómo levantar el servicio de MSGRPC desde una terminal y en otra, cómo establecer la conexión desde Python utilizando la librería.



```
(kali㉿kali)-[~]
$ msfconsole -q
msf6 > load msgrpc Pass=password
[*] MSGRPC Service: 127.0.0.1:55552
[*] MSGRPC Username: msf
[*] MSGRPC Password: password
[*] Successfully loaded plugin: msgrpc
msf6 >

(kali㉿kali)-[~]
$ python3
Python 3.9.2 (default, Feb 28 2021, 17:03:44)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from pymetasploit3.msfrpc import *
>>> client = MsfRpcClient('password', port=55552)
>>> [m for m in dir(client) if not m.startswith('_')]
['add_perm_token', 'auth', 'authenticated', 'call', 'consoles', 'core', 'db', 'encoding', 'headers', 'host', 'j
'port', 'post_request', 'sessions', 'ssl', 'token', 'uri']
>>>
```

Figura 2.1: Conexión entre Metasploit Framework y Python

En la imagen anterior se crea un objeto MsfRpcClient que es el que permite establecer la conexión con el servicio MSGRPC y que cuenta con todos los elementos necesarios para interactuar con el framework. Tal como se aprecia en la imagen, se obtiene un listado de todos los elementos disponibles en el objeto, cada uno de ellos permite acceder a características concretas de Metasploit Framework.

En el script **2-msf.py** se puede apreciar un procedimiento sencillo contra un sistema vulnerable, en este caso concreto se ha utilizado la máquina virtual vulnerable por diseño Metasploitable 2 (<https://sourceforge.net/projects/metasploitable/files/Metasploitable2>)

Dicha máquina tiene varios servicios con vulnerabilidades conocidas y todos ellos se pueden explotar utilizando Metasploit Framework. En el script indicado anteriormente se intenta explotar un servicio SMB sin actualizar cuyo módulo de explotación se encuentra disponible en Metasploit Framework. Como se ha dicho, antes de ejecutar este script hay que iniciar el servicio MSGRPC y además, es necesario cambiar en el script la IP correspondiente a la propiedad LHOST. Debe tener la dirección IP de la máquina donde se está ejecutando Metasploit Framework.

3 PUNTOS CLAVE

- | En la etapa de recolección activa, es de vital importancia conocer detalles concretos sobre el sistema objetivo. De esta manera será posible detectar y explotar vulnerabilidades más fácilmente.
- | Tal como se ha visto en este módulo, crear una rutina de “banner grabbing” no requiere el uso de librerías adicionales, solo es necesario utilizar los componentes disponibles en la instalación de Python.
- | Metasploit Framework es una herramienta ampliamente extendida en procesos de pentesting y hacking. Gracias al uso del servidor MSGRPC que trae incorporado, es posible integrar esta herramienta con cualquier script en Python.
- | La librería pymetasploit3 permite realizar una conexión con un servidor MSGRPC de Metasploit Framework y facilita su automatización.
- | El cliente disponible en pymetasploit3 es ligero y fácil de utilizar. Es posible indicar las instrucciones de Metasploit Framework directamente en el script sin necesidad de conocer la API del framework.

